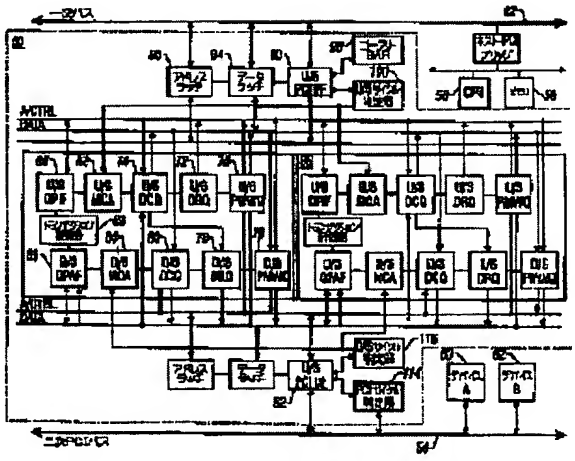
DELIVERY TRANSACTION WITH DATA BUS IN COMPUTER SYSTEM

Publication number: JP10301893
Publication date: 1998-11-13
Inventor: HAUSAUER BRIAN S
Applicant: COMPAQ COMPUTER CORP
Classification:
- international: ***G06F13/40;*** **G06F13/40;** (IPC1-7): G06F13/36
- european: G06F13/40D5S4
Application number: JP19970370468 19971226
Priority number(s): US19960777575 19961231

Also published as:
EP0851361 (A
US6138192 (A
EP0851361 (B

Abstract of **JP10301893**

PROBLEM TO BE SOLVED: To strictly perform ordering only relating to another transaction and to mitigate a transaction order rule by completing the transaction between a first device and a second device regardless of whether entrance to a transaction bridge device is performed before or after the transaction between the first device and a third device. SOLUTION: A computer system is provided with a PCI-PCI bridge bus 50 (PPB or 'bridge') respectively connected to primary and secondary PCI buses 52 and 54 corresponding to the mitigated transaction ordering rule. Especially, the bridge bus 50 performs the operation of ordering only relating to the other transaction between the two devices regardless of generation order relation among the transaction generated between a CPU 58 and a PCI device 62 ('device B') and the transactions for the transaction between the CPU 58 and the PCI device 60 ('device A').



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－301893

(43)公開日　平成10年(1998)11月13日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　13/36	３１０	Ｇ０６Ｆ　13/36	３１０Ｆ
			３１０Ｃ

審査請求　未請求　請求項の数25　ＯＬ　外国語出願　（全 40 頁）

(21)出願番号　特願平9－370468

(22)出願日　平成９年(1997)12月26日

(31)優先権主張番号　７７７５７５

(32)優先日　1996年12月31日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591030868
コンパック・コンピューター・コーポレーション
COMPAQ COMPUTER CORPORATION
アメリカ合衆国テキサス州77070，ヒューストン，ステイト・ハイウェイ　249，20555

(72)発明者　ブライアン・エス・ハウザー
アメリカ合衆国テキサス州77379，スプリング，ティンバーワーク　16914

(74)代理人　弁理士　社本　一夫　（外５名）

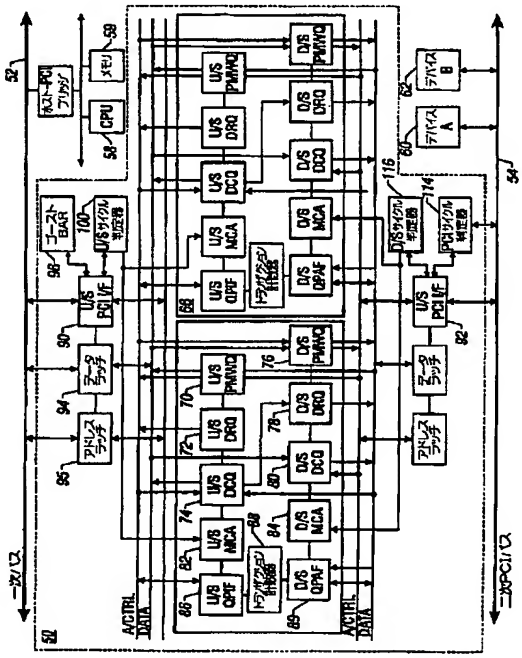
最終頁に続く

(54)【発明の名称】　コンピュータシステムにおけるデータバス間の送達トランザクション

(57)【要約】　（修正有）

【課題】２つのデバイス間のトランザクションが同じ２つのデバイス間の他のトランザクションに関してのみ厳格に順序付けされるようにするためにトランザクション順序規則を緩和する。

【解決手段】ブリッジデバイス５０を介して処理し得る各対のデバイス６０，６２に対して、この対のデバイス間のトランザクションを完了するのを助ける専用記憶領域６４，６６を含んでいるコンピュータシステムにおける２つのデータバス上のデバイス間のデータトランザクションの送達のためのブリッジデバイス。ブリッジデバイスは、また、ある専用記憶領域におけるトランザクションが、別の専用記憶領域における前に発行されたトランザクションの完了に拘わりなく完了出来るようにするコントローラを含む。



1

【特許請求の範囲】

【請求項１】　コンピュータシステムにおいて、第一データバス上の第一デバイス、第二データバス上の第二及び第三デバイス、及び上記第一及び第二データバス間に介在しているブリッジデバイスであって、上記第一及び第三デバイス間のトランザクションの前或いは後で上記第一デバイスと上記第二デバイス間のトランザクションが上記ブリッジデバイスにエンタしたか否かに拘わらず上記トランザクションを完了するように構成されているブリッジデバイスを含むことを特徴とするコンピュータシステム。

【請求項２】　上記ブリッジデバイスが上記第一及び上記第二デバイス間のトランザクションの発生順順序を保存することを特徴とする請求項１のコンピュータシステム。

【請求項３】　上記ブリッジデバイスが上記第一デバイスと上記第二デバイス間の遅延されたトランザクションを、それが上記第一及び第二デバイス間の別のトランザクションの完了を妨げている時に、完了を加速することを試みることを特徴とする請求項１のコンピュータシステム。

【請求項４】　上記の別のトランザクションがＣＰＵからのダウンストリーム読出しトランザクションを含むことを特徴とする請求項３のコンピュータシステム。

【請求項５】　上記第二データバス上のデバイスのどれが上記第一デバイスによって開始されたトランザクションの目標となるかを決定することを助けるゴーストベースアドレスレジスタを更に含むことを特徴とする請求項１のコンピュータシステム。

【請求項６】　上記第二又は第三デバイスのどちらかへの上記第二データバスの制御を許可する許可信号を更に含むことを特徴とする請求項１のコンピュータシステム。

【請求項７】　上記ブリッジデバイスが、上記第二又は第三デバイスが上記第二データバス上のトランザクションを開始したか否かを上記許可信号を用いて決定するように構成されていることを特徴とする請求項６のコンピュータシステム。

【請求項８】　上記データバスの１つがＰＣＩバスを含むことを特徴とする請求項１のコンピュータシステム。

【請求項９】　上記データバスの各々がＰＣＩバスを含むことを特徴とする請求項１のコンピュータシステム。

【請求項１０】　第一データバス上のデバイスと第二データバス上の他の２つのデバイスの各々との間での上記データバスを扱うブリッジデバイスを介しての２つのトランザクションの送達を制御するのに用いられる方法において、上記ブリッジデバイスに上記トランザクションを記憶する段階、及び上記トランザクションが上記ブリッジデバイスに記憶された発生順順序に拘わらず上記トランザクションの一方を送達する段階を含むことを特徴とする方法。

2

【請求項１１】　その後、他方のトランザクションを送達する段階を更に含むことを特徴とする請求項１０の方法。

【請求項１２】　上記第一データバス上のデバイスと上記他方のデバイスの一方との間で送達されるトランザクションの発生順順序を保存する段階を更に含むことを特徴とする請求項１０の方法。

【請求項１３】　上記第一データバス上のデバイスと上記他方デバイスの一方との間の遅延されたトランザクションの送達を、該遅延されたトランクザンクションが別のトランザクションを上記デバイス間で送達されないように妨げている時に、加速することを試みる段階を更に含むことを特徴とする請求項１０の方法。

【請求項１４】　上記他方のトランザクションがＣＰＵからのダウンストリーム読出しトランザクションを含むことを特徴とする請求項１３の方法。

【請求項１５】　上記バスの一つがＰＣＩバスを含むことを特徴とする請求項１０の方法。

【請求項１６】　上記バスの各々がＰＣＩバスを含むことを特徴とする請求項１０の方法。

【請求項１７】　コンピュータシステムにおいて、一次ＰＣＩバス上のＣＰＵと、二次ＰＣＩバス上の２つのＰＣＩデバイスと、上記一次及び二次バス間に介在しているＰＣＩ－ＰＣＩブリッジデバイスと、一方が上記ＣＰＵと上記ＰＣＩデバイスの各々との間で送達される２つのトランザクションを記憶する記憶領域と、上記トランザクションが上記記憶領域に記憶された発生順順序に拘わらず上記トランザクションの一方を送達のために選択する調停回路と、を含むＰＣＩ－ＰＣＩブリッジデバイスを含むことを特徴とするコンピュータシステム。

【請求項１８】　上記調停回路が、上記一方のトランザクションを選択した後に他方のトランザクションを選択するように構成されていることを特徴とする請求項１７のコンピュータシステム。

【請求項１９】　上記記憶領域が、一方が上記ＣＰＵと上記ＰＣＩデバイスの１つとの間で生じたトランザクションを記憶し、他方が上記ＣＰＵと他方のＰＣＩデバイス間で生じたトランザクションを記憶する２つの待ち行列ブロックを含むことを特徴とする請求項１７のコンピュータシステム。

【請求項２０】　上記記憶領域が、上記ＣＰＵから上記ＰＣＩデバイスの第一のものに流れるデータを記憶する１つのバッファ並びに上記第一ＰＣＩデバイスから上記ＣＰＵに流れるデータを記憶する別のバッファを有する第一組のバッファ、及び上記ＣＰＵから上記ＰＣＩデバイスの第二のものに流れるデータを記憶する１つのバッファ並びに上記第二ＰＣＩデバイスから上記ＣＰＵに流れるデータを記憶する別のバッファを有する第二組のバッファを含むことを特徴とする請求項１７のコンピュー

タシステム。

【請求項２１】　コンピュータシステムにおける２つのデータバス上のデバイス間でデータトランザクションを送達するためのブリッジデバイスにおいて、上記ブリッジデバイスを介して処理し得る各対のデバイスに対して、上記対のデバイス間のトランザクションを完了するのを助ける専用記憶領域、及び１つの専用記憶領域におけるトランザクションを別の専用記憶領域における前に発行されたトランザクションの完了に拘わらず完了せしめる制御装置を含むことを特徴とするブリッジデバイス。

【請求項２２】　各々の専用記憶領域が上記ブリッジデバイスを介して特定の方向に流れるデータに割り当てられた部分を含むことを特徴とする請求項２１のブリッジデバイス。

【請求項２３】　各々の専用記憶領域が上記ブリッジデバイスを介して上記と反対の方向に流れるデータに割り当てられた別の部分を含むことを特徴とする請求項２２のブリッジデバイス。

【請求項２４】　上記専用記憶領域が上記ブリッジを交差し得る各々の種類のトランザクションのための専用バッファを含むことを特徴とする請求項２１のブリッジデバイス。

【請求項２５】　上記トランザクションのタイプが、ポストされた書込み、遅延された要求、及び遅延された完了を含むことを特徴とする請求項２４のブリッジデバイス。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、コンピュータシステムにおけるデータバス間の送達トランザクションに関するものである。

【０００２】

【従来の技術】コンピュータシステムはコンピュータシステムの諸成分どうしが通信出来るようにする１つ又はそれ以上のデータバスを含んでいることが多い。例えば、１つの共通型のデータバスは、コンピュータシステムのＣＰＵ／主メモリと周辺成分間で特別な通信プロトコルを行う周辺成分インターフェース（ＰＣＩ）バスであり、該ＰＣＩには小コンピュータシステムインターフェース（ＳＣＳＩ）デバイスやネットワークインターフェースカード（ＮＩＣｓ）が含まれる。システムメモリ及びこの周辺成分（例えば、ＰＣＩデバイス）が別々のバス上に常駐している時は、これら２つのバス間のデータトランザクションの流れを管理するのにブリッジが必要となる。通常、システムメモリとＰＣＩデバイスは各々、ＰＣＩ－ＰＣＩブリッジによって結合されているＰＣＩバス上に、少なくとも間接的に常駐している。ＰＣＩバスアーキテクチャは、参照として含まれている、オレゴン州ポートランド在のＰＣＩスペシャルインタリストグループによる１９９５年６月発行のＰＣＩ局所的バス仕様（改定版２．１）（”ＰＣＩ仕様２．１）によって定義されている。ＰＣＩ－ＰＣＩブリッジアーキテクチャは、これも参照として含まれている、ＰＣＩスペシャルインタリストグループによる１９９４年４月発行のＰＣＩ－ＰＣＩブリッジアーキテクチャ仕様（改定版１．０（”ＰＣＩブリッジ仕様１．０）によって定義されている。

【０００３】ＰＣＩ仕様２．１及びＰＣＩブリッジ仕様１．０規格の下では、ＰＣＩブリッジは２つの型のトランザクションを支持している。即ち、目標バス（ｔａｒｇｅｔｂｕｓ）上で完了する前に開始バス（ｉｎｉｔｉａｔｉｎｇ　ｂｕｓ）上で完了するポストされたトランザクション（全てのメモり書込みサイクルを含む）、及び開始バス上で完了する前に目標バス上で完了する遅延トランザクション（メモリ読出し要求及びＩ／Ｏ及び構成読出し／書込み要求を含む）である。ＰＣＩ仕様２．１によると、ＰＣＩ－ＰＣＩブリッジは諸トランザクションが目標バス上で実行される順番を決定する時にポストされた書込みトランザクションを強く優先しなければならない。ＰＣＩ仕様２．１は、ロック・アップ状態（ｌｏｃｋ－ｕｐ）を防ぐために、コンピュータシステムはポストされた書込みサイクルに前に初期化された遅延要求サイクルを迂回させなければならないし、かつ遅延要求サイクルが前に初期化されたポストされた書込みサイクルを迂回しないように防がなければならないように要求している。この要求は、ポストされた書込みトランザクションと遅延された要求トランザクションが２つの異なった対のデバイスの間で生じても保持される。

【０００４】図１について説明する。通常のコンピュータシステムは、ホストバス１４に接続されている中央処理装置（ＣＰＵ）１０及び主メモリデバイス１２（メモリ制御装置を含む）を含んでいる。ホストバス１４はホストーＰＣＩブリッジデバイス１８を通してＰＣＩバス１６と通信している。ＰＣＩバス１６はＰＣＩ－ＰＣＩブリッジ２０を通して別のＰＣＩバス１９と交互に通信している。ＰＣＩバス１６は”一次”ＰＣＩバスとして知られており、ＰＣＩバス１９は”二次”ＰＣＩバスとして知られているが、これはＰＣＩバス１６がＰＣＩバス１９よりもホストバス１４に階層的に近接しているからである。

【０００５】ビデオカード２２、ＮＩＣ２４、及びＰＣＩオプションカード２６を含む幾つかの周辺デバイスは二次ＰＣＩバス１９に接続されている。各デバイスは二次ＰＣＩバス１９上の“スロット”に差し込まれてつながる。ＰＣＩオプションカード２６は、ＰＣＩ－ＰＣＩブリッジデバイス２８及びそれ自身のＰＣＩバス３０を含んでおり、このＰＣＩバス３０によって付加的なＰＣＩデバイスが二次ＰＣＩバス１９上のシングルスロット（ｓｉｎｇｌｅ　ｓｌｏｔ）に差し込まれてつながる。

１つ又はそれ以上のＰＣＩデバイス、例えばＳＣＳＩデバイス３２がＰＣＩオプションカード２６上に常駐し得る。ＰＣＩオプションカード２６は、２つ以上のＰＣＩデバイスがその局所的ＰＣＩバス３０に接続されている時、“多条（ｍｕｌｔｉ－ｔｈｒｅａｄｅｄ）”デバイス（即ち、多数の遅延されたトランザクションを同時に維持することが出来るデバイス）として作動する。

【０００６】ＣＰＵによって開始されたトランザクション及び二次ＰＣＩバス１９上のデバイスを目標とすることは“ダウンストリーム”トランザクションとして知られており、二次バス１９上のデバイスによって開始されたトランザクション及び主メモリ１２を目標とすることは”“アップストリーム”トランザクションとして知られている。二次バス１９上のデバイスによって開始されるトランザクション及び二次バス１９上の別のデバイスを目標とすることは”対等”トランザクションとして知られている。対等トランザクションは本明細書では扱わない。

【０００７】

【発明が解決しようとする課題】ＰＣＩ仕様２．１によると、一次又は二次ＰＣＩバス上で開始された且つＰＣＩ－ＰＣＩブリッジデバイス２０においてポストされた如何なる書込みトランザクションも、次に発行されたトランザクションのどれかが目標バス上で完了され得る前に目標バス上で完了されなければならない。加うるに、遅延された読出しトランザクションがバス上で開始される前に開始バス上で完了する任意のトランザクションも、その遅延された読出しトランザクションが目標バス上で完了される前に目標バス上で完了されなければならない。同様に、遅延された読出しトランザクションが開始バス上で完了した後で開始バス上で完了される任意のトランザクションも、目標バス上の遅延された読出しトランザクションの後に完了されなければならない。それ故、ＳＣＳＩデバイス２２によってＰＣＩ－ＰＣＩブリッジ２０においてポストされたメモリ書込みトランザクションは、ＮＩＣ２４からの後に発行される遅延された読出し要求が完了される前に一次ＰＣＩバス１６上で完了されなければならないし、ポストされたメモリ書込みトランザクションは、ＮＩＣ２４からの任意の前に発行された遅延された読出しトランザクションに関連した事前取出しデータを無効にしなければならない（即ち、メモリ書込みトランザクションがポストされた時、事前取出しデータはＰＣＩ－ＰＣＩブリッジ２０から流れなければならない）。

【０００８】

【課題を解決するための手段】第一の特徴によると、本発明は、第一データバス上の第一デバイス、第二データバス上の第二及び第三デバイス、第一及び第二データバス間に介在されたブリッジデバイスであって第一デバイスと第三デバイスとの間のトランザクションの前或いは後にトランザクションがブリッジデバイスにエンターしたかに拘わらず第一デバイスと第二デバイスとの間のトランザクションを完了するように構成されているブリッジデバイスを有するコンピュータシステムを特徴としている。

【０００９】本発明の実施態様は以下の特徴の１つ又はそれ以上を含んでいる。ブリッジデバイスは、第一デバイスと第二デバイスとの間のトランザクションの発生順順序を保存し得る。ブリッジデバイスは、遅延されたトランザクションが第一及び第二デバイスとの間の別のトランザクション（例えば、ＣＰＵからのダウンストリーム読出しトランザクション）の完了を防いでいる時に第一及び第二デバイス間の遅延されたトランザクションの完了を加速するように試み得る。コンピュータシステムは、第二データバス上のどのデバイスが第一デバイスによって開始されたトランザクションの目標であるかを決定することを助けるゴーストベースアドレスレジスタ（ｇｈｏｓｔ ｂａｓｅ ａｄｄｒｅｓｓ ｒｅｇｉｓｔｅｒ）を含み得る。

【００１０】コンピュータシステムはまた、第二或いは第三デバイスのどちらかへの第二データバスの制御を許可する許可信号を含み得る。そして、ブリッジデバイスは第二或いは第三デバイスが第二データバス上のトランザクションを開始したかを決定するのにこの許可信号を使用するように構成されている。これらのデータバスの一方または両方はＰＣＩバスであり得る。

【００１１】別の特徴によると、本発明は、データバスを取り扱うブリッジデバイスを通る第一データバス上のデバイスと第二データバス上の２つの他のデバイスの各々間の２つのトランザクションの送達を制御するのに用いられる方法を特徴とする。これらのトランザクションは先ず、ブリッジデバイスに記憶され、次に、これらのトランザクションの一方が、これらのトランザクションがブリッジデバイスに記憶された発生順順序に拘わらず送達される。次に他方のトランザクションが送達され得る。

【００１２】本発明の実施態様は以下の特徴の１つ又はそれ以上を含み得る。第一データバス上のデバイスと他方のデバイスの１つとの間で送達されるトランザクションの発生順順序は保存され得る。第一データバス上のデバイスと他方のデバイスの１つとの間の遅延されたトランザクションの送達は、遅延されたトランザクションが別のトランザクション（即ち、ＣＰＵからのダウンストリーム読出しトランザクション）がこれらのデバイス間で送達されないように防いでいる時に加速され得る。これらのバスの一方又は両方はＰＣＩバスであり得る。

【００１３】別の特徴によると、本発明は、一次ＰＣＩバス上のＣＰＵ、第二ＰＣＩバス上の２つのＰＣＩデバイス、及び一次及び二次バス間に介在するＰＣＩ－ＰＣＩブリッジデバイスを有するコンピュータシステムを特

徴とする。このブリッジデバイスは、一方がＣＰＵとＰＣＩデバイスの各々間で送達される２つのトランザクションを記憶する記憶領域、並びにこれらのトランザクションが記憶領域に記憶された発生順順序に拘わらずこれらのトランザクションの１つを送達用に選択する仲介回路を含んでいる。

【００１４】本発明の実施態様は以下の特徴の１つ又はそれ以上を含み得る。仲介回路は１つのトランザクションを選択した後に他方のトランザクションを選択するように構成され得る。記憶領域は、その一方がＣＰＵとＰＣＩデバイスの１つとの間に生じるトランザクションを記憶し、他方がＣＰＵと他方のＰＣＩデバイスとの間に生じるトランザクションを記憶する２つの待ち行列を含み得る。記憶領域は、ＣＰＵからＰＣＩデバイスの第一の１つに流れるデータを記憶する１つのバッファ並びに第一ＰＣＩデバイスからＣＰＵに流れるデータを記憶する別のバッファを有する第一組のバッファを含み得る。記憶領域はまた、ＣＰＵからＰＣＩデバイスの第二の１つに流れるデータを記憶する１つのバッファ並びに第二ＰＣＩデバイスからＣＰＵに流れるデータを記憶する別のバッファを有する第二組のバッファを含み得る。

【００１５】別の特徴によると、本発明は、コンピュータシステムにおける２つのデータバス上のデバイス間でデータトランザクションを送達するためのブリッジデバイスを特徴とする。

【００１６】ブリッジデバイスを介して処理し得る各々の対のデバイスに関して、ブリッジデバイスは、この対のデバイス間のトランザクションを完了することを助ける専用の記憶領域を含んでいる。ブリッジデバイスは又、１つの専用記憶領域が前に発行されたトランザクションの別の専用記憶領域における完了に拘わらず完了せしめる制御装置を含んでいる。

【００１７】本発明の実施態様は以下の特徴の１つ又はそれ以上を含み得る。各専用記憶領域は、ブリッジデバイスを介した特定の方向に流れるデータに割り当てられた部分を含み得る。そして、各専用記憶領域は又、ブリッジデバイスを介した反対方向に流れるデータに割り当てられた部分を含み得る。この専用記憶領域はブリッジを交差し得る各種のトランザクションに対する専用バッファを含み得る。これらのトランザクション種類は、ポストされた書込み、遅延された要求、及び遅延された完了を含み得る。

【００１８】本発明の実施態様の利点には以下のものが含まれる。２つのデバイス間のトランザクションが同じ２つのデバイス間の他のトランザクションに関してのみ厳格に順序付けされるようにするためにトランザクション順序規則を緩和し得る。例えば、遅延されたトランザクションアーキテクチャにおいて、ＣＰＵと第一ＰＣＩデバイスとの間の遅延された読出しトランザクションをＣＰＵと別のＰＣＩデバイスとの間の前に発行されたポストされた書込みトランザクションの前に完了され得る。加うるに、ＣＰＵと第一ＰＣＩデバイスとの間の前に発行された遅延された読出しトランザクションに関連した事前取出しデータは、ＣＰＵと他方のＰＣＩデバイスがポストされた時には無効にする必要がない。それ故、ＰＣＩブリッジに依存するコンピュータシステムはより効率的に作動し得る。何となれば、遅延されたトランザクション中に送達されるデータは頻繁に破棄されないからである。

【００１９】

【実施例】図２について説明する。コンピュータシステム５０は、緩和されたトランザクション順序付け規則に従う一次及び二次ＰＣＩバス５２及び５４にそれぞれ接続しているＰＣＩ－ＰＣＩブリッジデバイス５０（ＰＰＢ又は“ブリッジ”）を含む。特に、ブリッジ５０は、ＣＰＵ５８とＰＣＩデバイス６０（“デバイスＡ”）との間のトランザクションが、これらのトランザクションとＣＰＵ５８とＰＣＩデバイス６２（“デバイスＢ”）との間で生じるトランザクションとの間の発生順関係に拘わらず、これら２つのデバイス間の他方のトランザクションに関してのみ順序付けされるように作動する。

【００２０】ブリッジ５０は、２つの同等の組のバッファ６４及び６６、即ち、ブリッジ５０を介して処理し得る各ＰＣＩデバイス“対”に対して“待ち行列ブロック”を維持することによって緩和されたトランザクション順序付けを達成する。例えば、待ち行列ブロック６４は一時的にＣＰＵ５８とデバイスＡ６０との間に生じるトランザクションを記憶し、待ち行列ブロック６６はＣＰＵ５８とデバイスＢ６２間に生じるトランザクションを一時的に記憶する。ブリッジ５０は二次ＰＣＩバス５４上の各々の他のＰＣＩデバイスに対する付加的な待ち行列ブロックを含む。ブリッジデバイス５０は、ＣＰＵ５８及びメモリ５９を一次ＰＣＩバス５２上の唯１つのデバイスとして処理する。待ち行列ブロック６４及び６６におけるバッファは、二次ＰＣＩスロットに永久的に割り当てられる必要がないが、その代わり、動的に、例えば、システム開始に割り当てられ得る。

【００２１】各待ち行列ブロック６４及び６６は、一次ＰＣＩバス５２と二次ＰＣＩバス５４上のデバイス６０及び６２間に流れるトランザクションを管理する。待ち行列ブロック６４及び６６は同様であり、待ち行列ブロック６４のみを以下に説明する。待ち行列ブロック６６（及びブリッジデバイス５２に存在し得る他の何らかの待ち行列ブロック）も同様に機能する。各待ち行列ブロックは、Ｊｏｈｎ Ｍａｒｃｌａｒｅｎ とＡｌａｎ Ｇｏｏｄｒｕｍｔとにより、１９９６年６月５日に出願され、本願発明の参照とされる、米国特許第０８／６５５、２５４号に開示され記載されている待ち行列に類似している。

【００２２】各待ち行列ブロック６４は、（二次ＰＣＩ

バス５４上の対応するＰＣＩデバイス６０からＣＰＵ５８に）アップストリームに流れる情報を記憶する３つのアップストリーム待ち行列(upstream queue)７０、７２、及び７４、並びに、（ＣＰＵ５８から対応するＰＣＩデバイス６０に）ダウンストリームに流れる情報を記憶する３つのダウンストリーム待ち行列(downstream queue)７６、７８、及び８０を含んでいる。待ち行列ブロック６４は又、以下に述べるように、アップストリーム又はダウンストリーム待ち行列のどちらからか出る情報の流れを支配する２つのサイクルアービタ８２及び８４並びに２つの待ち行列ブロックーＰＣＩ（ＱＰＩＦ）インターフェース８６及び８９を含んでいる。

【００２３】第一ダウンストリーム待ち行列、即ちポストされたメモリ書込み待ち行列７６は、二次バス５４上の各サイクルを実施するのに必要な全ての情報と共に一次バス５２上のＣＰＵ５８によって発行されるポストされたメモリ書込みサイクルを記憶する。ＰＭＷＱ７６は、各々が8本までのキャッシュライン（２５６バイト）のデータを含む１つのポストされたメモリ書込みトランザクションを保持する４つのトランザクションバッファを有する。ダウンストリームＰＭＷＱ７０は二次バス５４上のＰＣＩデバイス６０によって発行されたポストされたメモリ書込みサイクルを記憶し、ダウンストリームＰＭＷＱ７６と同様に機能する。これらのＰＭＷＱ７０及び７６は、１９９６年６月に出願された米国特許出願第０８／６５５、２５４号により詳細に記載されている。

【００２４】第二ダウンストリーム待ち行列、即ち、遅延された要求待ち行列（ＤＲＱ）７８は、二次バス５４上の各トランザクションを実施するのに必要な全ての情報と共に、一次バス５２上のＣＰＵ５８によって発行された遅延された要求トランザクション〔即ち、メモリ読出し（ＭＲ）、メモリ読出しライン（ＭＲＬ）、及びメモリ読出し多重（ＭＲＭ）要求等の遅延された読出し要求（ＤＲＲ）、並びに入力／出力（Ｉ／Ｏ）読出し／書込み及び構成（ｃｏｎｆｉｇ）読出し／書込み〕を記憶する。ＤＲＱ７８は、それぞれが遅延された書込みに対するデータの１つの二重ワード、即ち“ｄｗｏｒｄ”を保持することが出来る３つのトランザクションバッファを有している。同様にして、アップストリームＤＲＱ７２は、二次バス６４上のＰＣＩデバイス６０によって発行された遅延された要求トランザクションを記憶し、ダウンストリームと同様に機能する。ＤＲＱ７２及び７８は、１９９６年６月５日に出願された米国特許出願第０８／６５５、２５４号により詳細に記載されている。

【００２５】第三ダウンストリーム待ち行列、即ち、遅延された完了待ち行列（ＤＣＱ）８０は、ＰＣＩデバイス６０によって発生された遅延された要求トランザクションに応答してＣＰＵ５８によって与えられた遅延された完了情報を記憶する。遅延された読出し要求に対しては、対応する完了情報は、開始デバイスによって要求された読出しデータ及び読出し状態〔即ち、パリティエラー或いは目標アボート(target abort)のどちらかが生じたかの指示〕を含んでいる。遅延された書込みトランザクションに対して帰還した遅延された完了情報は、遅延された書込みに対してデータが何も帰還されないこと以外、遅延された読出し要求に対して帰還したそれと同じである。

【００２６】ＤＣＱ８０は、それぞれが唯一つの遅延された要求に対して８本までのキャッシュラインの完了ラインを保持することが出来る８個の完了バッファを有する。完了情報に加えて、各完了バッファは又、その情報を発生した遅延された要求の複写を記憶する。この遅延された要求の複写は、発行ＰＣＩデバイスによる後続の要求が記憶された要求の再試行となる時を決定するのに用いられる。後続の要求が記憶された要求に一致し、完了バッファが要求されたデータを含む場合、ＤＣＱ８０は完了データを要求デバイスに与える。

【００２７】同様にして、アップストリームＤＣＱ７４は、ＣＰＵ５８からの遅延された要求に応答してＰＣＩデバイス６０によって与えられる遅延された完了情報を記憶する。Ｉ／Ｏ及びｃｏｎｆｉｇ読出し／書込みがダウンストリームバス上にのみ生じるため、アップストリームＤＣＱ７４のみが、これらのトランザクションの１つに対応する遅延された完了情報を含み得る。

【００２８】ＤＣＱ７４又は８０の一方が、そのバッファの１つが要求デバイスに対して意図されているが、現在のトランザクションにおいて要求されるデータとは異なるデータを含むことを決定する場合、バッファがフラッシュされて要求マスタが古いデータ(stale ｄａｔａ）を受けないように防ぐことが出来る。バッファは、それが事前取出しデータ（即ち、要求デバイスがデータの幾つかを取り出した後にバッファに残されたデータ、或いはデバイスによって特別に要求されなかったデータ）を含んでいる時にはフラッシュ（ｆｌｕｓｈ）されるが、それが完了データ（即ち、それを取り出すために未だ帰還してないデバイスによって特別に要求されたデータ）を含んでいる時にはフラッシュされない。バッファが完了データを含み且つ要求デバイスがバッファを“ヒット（ｈｉｔ）”しないことの要求を発行している場合、ＤＣＱ１４４はこのデバイスを“多条”デバイス（即ち、一度に２つ以上のトランザクションを維持することが出来るデバイス）として標識を与え、この新しい要求に別の完了バッファを割り当てる。

【００２９】ＤＣＱ７４及び８０における各バッファはデータが通過すると幾つかの状態を通して進行する。バッファは、それがデータを何も含んでいない時は“空”状態にあり、それが開始デバイスによって特別に要求されたデータを含んでいる時は“完了”状態にあり、且つ、それが（例えば、開始デバイスがバッファから全て

の要求されたデータを取り出し且つその“フレーム”信号を主張し続けて、それがデータをもっと希望していることを指示した後）ブリッジデバイス５２によって特別に要求されなかったが事前取出しされたデータを含んでいる時は“事前取出し”状態にある。このＤＣＱ７４及び８０は、１９９６年６月５日に出願された米国特許出願第０８／６５５、２５４号により詳細に記載されている。

【００３０】待ち行列ブロック６４における各マスタサイクルアービタ（ＭＣＡ）８２及び８４は、ダウンストリーム待ち行列或いはアップストリーム待ち行列のどちらかにおけるポストされたメモリ書込み、遅延された要求、及び遅延された完了トランザクション間の順序付け制約を維持する。緩和された順序付け制約は、厳格な順序付けが同じ対のデバイスを伴うトランザクション間のみで維持される（即ち、ＣＰＵ５８とＰＣＩデバイス６０間のトランザクションがこれら２つのデバイス間の他方のトランザクションについてのみ順序付けされる）ことを除いて、ＰＣＩブリッジアーキテクチャ仕様の２．１版に記載されているものと同様である。これらの順序付け制約は、バスサイクルが強い書込み順序付けを維持すること及びデッドロックが生じないことを要求する。それ故、各ＭＣＡ８２又は８４は、対応するＰＭＷＱ７０又は７６におけるポストされたメモリ書込みトランザクション並びに対応するＤＲＱ７２又は７８における遅延された要求トランザクションが対応するＰＣＩバス５２又は５４上の実行される順序を決定する。各ＭＣＡ８２又は８４は又、対応するＤＣＱ７４又は８０に記憶された遅延された完了情報の使用可能度を制御する。これらの規則との追従を確実にするために、ダウンストリームＭＣＡ８４は各ポストされたメモリ書込みサイクルに前に発行された遅延された要求サイクルを迂回する機会を与え、一方、ダウンストリーム及びアップストリームＭＣＡ８４及び８２は両方ともそれぞれ、遅延された要求及び遅延された完了サイクルに前に発行されたポストされたメモリ書込みサイクルを迂回させない。

【００３１】図３についても説明する。各ＭＣＡ８２又は８４は、２つのトランザクション待ち行列、即ち、トランザクション実行待ち行列（ＴＲＱ）（即ち、トランザクション実施待ち行列）２００及びトランザクション命令待ち行列（ＴＯＱ）２０２を用いて、対応するＰＭＷＱ、ＤＲＱ、及びＤＣＱにおいて待ち行列されるサイクルを管理する。ＭＣＡ制御ブロック２０４はＰＭＷＱ、ＤＲＱ、及びＤＣＱからトランザクションを受け、実行命令を出力する。これらのトランザクションは、それぞれ、ＴＲＱ制御ブロック２０６及びＴＯＱ制御ブロック２０８によってＴＲＱ２００及びと２０２に出入りする。

【００３２】ＴＲＱ２００は、ＭＣＡがトランザクション実施命令を決定する時の待ち行列である。ＴＲＱ２００におけるトランザクションは、トランザクション順序付け規則を犯すことなく任意の順序でも実施され得るが、一旦、ポストされたメモリ書込みサイクルがＴＲＱ２００に置かれると、ポストされたメモリ書込みが取り除かれるまで、他のサイクルは何もＴＲＱ２００に置かれ得ない。ＴＲＱ２００におけるトランザクションは循環順序で試行され、全体的に、それらが受けられる順序で完了される。しかしながら、ＴＲＱ２００におけるトランザクションがＰＣＩバス上で再試される場合、ＭＣＡ８２又は８４はＰＣＩバス上で試行されるＴＲＱ２００における次のトランザクションを選択し得る。遅延された完了トランザクションは、これらがマスタサイクルでなくスレーブサイクルであるためにＴＲＱ２００に決して置かれることがない。更に、遅延された完了情報が、ＰＭＷＱにおいてポストされたメモリ書込みサイクルが保留中である場合にそれがＤＣＱにエンタするとすぐに要求デバイスに対して得られるようになるため、遅延された完了トランザクションは、ポストされたメモリ書込みサイクルがＴＲＱ２００において保留中の時にのみＴＯＱ２０２に置かれる。

【００３３】ＴＲＱ２００は一度に４つまでのトランザクションを保持する循環待ち行列である。ＭＣＡは常に少なくとも１つのポストされたメモリ書込みトランザクションを実行させて所要順序付け制約を保存することが出来なければならないため、ＴＲＱ２００は一度に３つ以上の遅延された要求トランザクションを保持することが決して出来ない。更に、ＴＲＱは一度に１つだけのポストされた書込みトランザクションを保持出来ないが、これは、ポストされた書込みは、他のポストされた書込みを含む何らかの後に開始されたトランザクションにバスされることがないからである。

【００３４】ＴＯＱ２０２は、ポストされたメモリ書込みトランザクションがＴＲＱ２００に置かれた後にブリッジによって受けられたトランザクションの履歴的順序を保持する先入れ先だし（ＦＩＦＯ）待ち行列である。全てのトランザクションが実行するには前に発行されたポストされたメモリ書込みを待機しなければならないため、ポストされたメモリ書込み、遅延された要求、及び遅延された完了トランザクションを含む全てのトランザクションは、ポストされたメモリ書込みがＴＲＱ２００において待ち行列される時にＴＯＱ２０２に置かれる。ＴＯＱ２０２におけるトランザクションは、ポストされたメモリ書込みトランザクションがＴＲＱ２００から取り除かれるまでＴＯＱ２０２に保持されなければならない。ＴＯＱ２０２は３個までのポストされたメモリ書込みトランザクション（４個目はＴＲＱ２００に記憶される）、３個の遅延された要求トランザクション、及び遅延された完了トランザクションを保持し得る。ＴＲＱ２００、ＴＯＱ２０２、並びに制御ブロック２０４、２０６、及び２０８を含むＭＣＡ８２及び８４は、１９９６

年６月５日出願の米国特許出願第０８／６５５、２５４号により詳細に記載されている。

【００３５】アップストリーム待ち行列－ＰＣＩインターフェース（ＱＰＩＦ）８６は、アップストリーム待ち行列７０、７２、及び７４から一次ＰＣＩバス５２に且つ一次ＰＣＩバス５２からダウンストリームＤＣＱ８０に流れるトランザクションを管理する。ＱＰＩＦ８６は、“マスタ”モードに入って、一次バス５２上のＰＭＷＱ７０及びＤＲＱ７２からポストされたメモリ書込み及び遅延された要求トランザクションを実行する。ＱＰＩＦ８６は“スレーブ”モードに入り、アップストリームＤＣＱ７４からのデータをＣＰＵ５８に与え、一次ＰＣＩバス５２からのトランザクションをダウンストリームＤＣＱ８０に送る。

【００３６】ＱＰＩＦ８６がバス５２からポストされた書込みトランザクションを受けると、一群のトランザクション計数器８８の対応する１つがダウンストリームＰＭＷＱ７６が一杯でないことを指示している場合、このトランザクションをダウンストリームＰＭＷＱ７６に送る。ＱＰＩＦ８６が遅延された要求を受けると、それは先ず、この要求をＤＣＱ７４に送り、このトランザクションが既にアップストリームＤＣＱ７４に置かれているか否かを、そうである場合、対応する遅延された完了情報がＰＣＩデバイス６０によって帰還されているか否かを決定する。遅延された完了情報がアップストリームＤＣＱ７４にある場合、情報はＣＰＵ５８に与えられ、このトランザクションが終了する。この要求が既に待ち行列されているが、遅延された完了情報が帰還されていない場合、ＣＰＵ５８は再試され、このトランザクションは一次ＰＣＩバス５２上で終了する。このトランザクションが未だ待ち行列されていなく、トランザクション計数器８８の対応する１つがダウンストリームＤＲＱ７８が一杯でないことを指示する限り、アップストリームＤＣＱ７４はトランザクションに対する完了バッファを保存し、ＱＰＩＦ８６はこのトランザクションをダウンストリームＤＲＱ７８に送る。ダウンストリームＱＰＩＦ８９はアップストリームＱＰＩＦ８６と同様に機能する。ＱＰＩＦ８６及び８９は１９９６ね６月５日に出願された米国特許出願第０８／６５５、２５４号により詳細に記載されている。

【００３７】トランザクション計数器８８は、アップストリーム及びダウンストリーム待ち行列にそれぞれ待ち行列されているトランザクションの数の計数を維持する。ポストされたメモリ書込み（ＰＭＷ）計数器は、対応するポストされたメモリ書込み待ち行列に保持されたＰＭＷトランザクションの数を指示する。このＰＭＷ計数器は、ＰＭＷトランザクションが対応するＰＭＷＱに送られる度に増分する。この計数器は、ＰＭＷサイクルが対応ＰＣＩバス上で完了される度に減分する。同様に、遅延された要求（ＤＲ）計数器は対応する遅延された要求待ち行列に保持されたＤＲトランザクションの数を計数する。遅延された完了（ＤＣ）計数器は対応するマスタサイクルアービタにおいて待ち行列されている遅延された完了の数を計数する。これらのトランザクション計数器は、１９９６年６月５日に出願された米国特許出願第０８／６５５、２５４号により詳細に記載されている。

【００３８】ブリッジデバイス５２は、アップストリームＰＣＩインターフェースデバイス９０及びダウンストリームＰＣＩインターフェースデバイス９２を含んでいる。アップストリームＰＣＩインターフェース９０は、ブリッジデバイス５０を一次バス５２上のデバイス（即ち、ＣＰＵ５８及び主メモリ５９）と通信せしめ、ダウンストリームＰＣＩインターフェース９２は、ブリッジデバイス５０を二次バス５４上のデバイス（即ち、デバイスＡ６０及びデバイスＢ６０）と通信せしめる。

【００３９】ＣＰＵが一次バスの制御を行い且つトランザクションを開始する時に、アップストリームＰＣＩインターフェース９０はＰＣＩスレーブデバイスとなる。インターフェース９０はＰＣＩ命令、アドレス、及びデータをデータ及びアドレスラッチ９４及び９５にラッチする。そしてインターフェース９０はダウンストリームＰＣＩデバイス６０及び６２のどちらかがトランザクションの目標となるかを決定する。このようにするために、インターフェースはＣＰＵ５８によって与えられたメモリアドレスをブリッジデバイス５０におけるゴーストベースアドレスレジスタ（ゴーストＢＡＲ）９６におけるアドレスと比較する〔ゴーストＢＡＲ９６は、開始におけるＰＯＳＴによって形成されるベースアドレスレジスタ（ＢＡＲ）を詮索することによって形成される。各対のＢＡＲはＰＣＩスロットをメモリのブロックにマッピングし、ゴーストＢＡＲはこれらのマッピングの複写を含んでいる〕。

【００４０】ＰＣＩインターフェース９０は、ＣＰＵ５８は全てのダウンストリームトランザクションのマスタであることを仮定する。

【００４１】図４Ａについても説明する。アップストリームサイクルアービタ１００は、“レベルＩ”ラウンドロビン・スキーム１０５を用いて、アップストリームトランザクションが一次バス５２上で実行されるべき待ち行列ブロック６４又は６６を選択する。アービタ（ａｒｂｉｔｅｒ）１００は先ず、デバイスＡ６０（状態１０２）に対応する待ち行列ブロック６４を選択する。アービタ１００は次に、デバイスＢ６２に対応する待ち行列６６を選択し、他の任意のＰＣＩデバイスが二次バス５４に接続されている場合、アービタ１００は、これらのデバイスに対応する待ち行列を漸次に選択する。待ち行列ブロック６４及び６６の一方が実行すべきトランザクションを含んでいない場合、アービタ１００はそのラウンドロビン・スキーム１０５において対応する状態をス

キップする。選択された待ち行列ブロックからのトランザクションが再試される場合、アービタ１００は、ラウンドロビン・スキーム１０５において次の状態に移行する。

【００４２】図４Ｂについて説明する。ＰＣＩ仕様２．１は一次及び二次バスが両方ともトランザクションを同時に開始すると、アービタ１００はダウンストリームトランザクションに優先順位を与えなければならないことを規定する“ブリッジインターフェース優先”規則を含んでいる。しかし、書込みポストを可能にする幾つかのブリッジにおいては、この要求は、ダウンストリームデバイスがＣＰＵがダウンストリームデバイスに読出し要求を発行するのと同時に書込みをポストする場合にデッドロックに至り得る。この状態において、ブリッジ５０は、同じデバイスに対する任意の遅延された要求を完了する前にポストされた書込みを完了し且つダウンストリームマスタによって同時に発行された任意のトランザクションを完了する前にダウンストリーム読出しを完了することの両方を指示される。

【００４３】デッドロックが生じるのを防ぐために、幾つかの実施態様におけるアップストリームサイクルアービタは、この条件が生じた場合、“レベルII”調停スキーム１０８に切り替わる必要があり得る。レベルＩＩスキーム１０８において、アービタ１００は、このトランザクションをより迅速に完了しようとして、同時に発行されたポストされた書込みトランザクションの優先順位を上げる（状態１１０）。アップストリームインターフェース９０が、一次バス５２上のポストされた書込みトランザクションを実行することを試行した後、アービタ１００は図４Ａの“レベルＩ”調停スキーム１０５に帰還して次に実行するトランザクションを選択する。ポストされた書込みトランザクションが一次バス５２上で首尾良く実行しなかった場合、アービタ１００は、レベルＩスキーム１０５からのトランザクションを選択した後、レベルＩＩ調停スキーム１０８に帰還する（即ち、ポストされた書込みトランザクションが首尾良く完了するまで、アービタ１００はレベルＩＩスキーム１０８に継続する）。

【００４４】図２について再び説明する。ダウンストリームＰＣＩインターフェース９２は、幾つかの例外はあるが、アップストリームＰＣＩインターフェース９０と同様に作動する。第一に、アップストリームサイクルアービタ１００と異なり、ダウンストリームサイクルアービタ１１６は２レベル調停スキームを用いる必要は絶対にない。これは、ダウンストリームＰＣＩインターフェース９２が、”ブリッジインターフェース優先”規則に従わないからである。第二に、ダウンストリームインターフェース９２は常に、ＣＰＵ５８は目標デバイスであり且つＰＣＩデバイス６０又は６２のどちらがマスタであるかを決定するのにゴーストＢＡＲ９６に依存する必要がないと仮定している。その代わり、ダウンストリームインターフェース９２はＰＣＩアービタ１１４におけるＲＥＱ及びＧＮＴラインを監視して、二次ＰＣＩバス５４上のデバイス６０及び６２のどちらがマスタであるかを、即ち、待ち行列ブロック６４又は６６のどちらがトランザクションを受けるかを決定する。

【００４５】他の実施態様は、特許請求の範囲内にある。例えば、図５を参照すると、ＰＣＩ－ＰＣＩブリッジデバイスは、対当たり２組の単一方向待ち行列の代わりに、各々のＰＣＩデバイス対に対して１組の双方向トランザクション待ち行列１２０ａ、１２０ｂを含み得る。本発明は又、非ＰＣＩアーキテクチャ及びＰＣＩ仕様２．１ａ或いはＰＣＩブリッジ仕様１．０に従わないＰＣＩアーキテクチャの形で実施され得る。

【図面の簡単な説明】

【図１】図１は、コンピュータシステムのブロック図である。

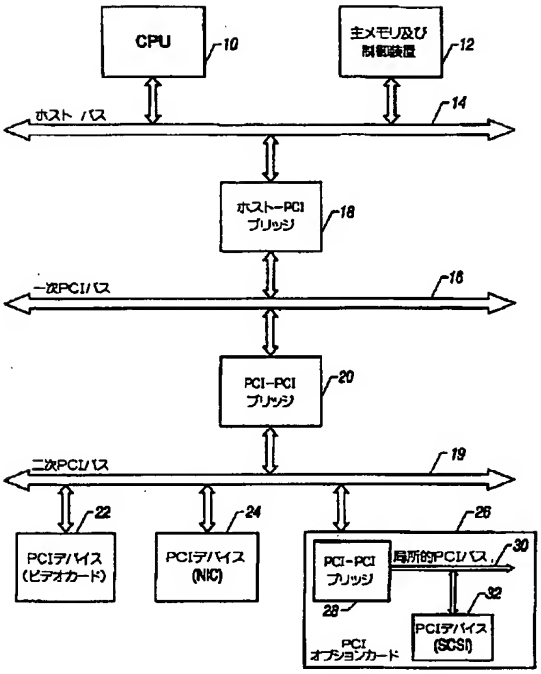
【図２】図２は、緩和されたトランザクション順序規則を実施するブリッジによってデータバスが接続されるコンピュータシステムのブロック図である。

【図３】図３は、トランザクションの適切な順序付けを確実にするマスタサイクルアービタのブロック図である。

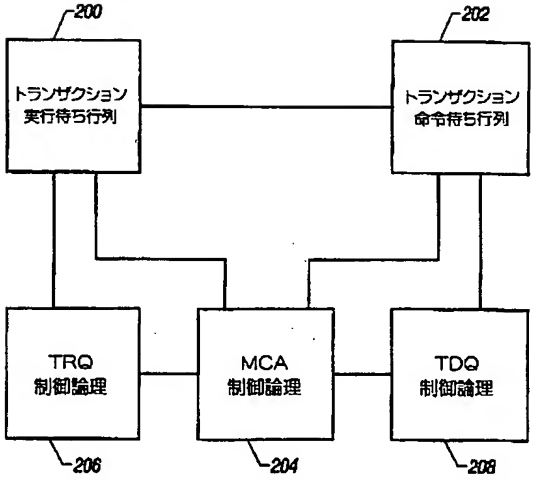
【図４】図４は、図４Ａ及び図４Ｂを含み、図４Ａ及び図４Ｂは調停スキームの状態遷移図である。

【図５】図５は、代替ブリッジアーキテクチャのブロック図である。

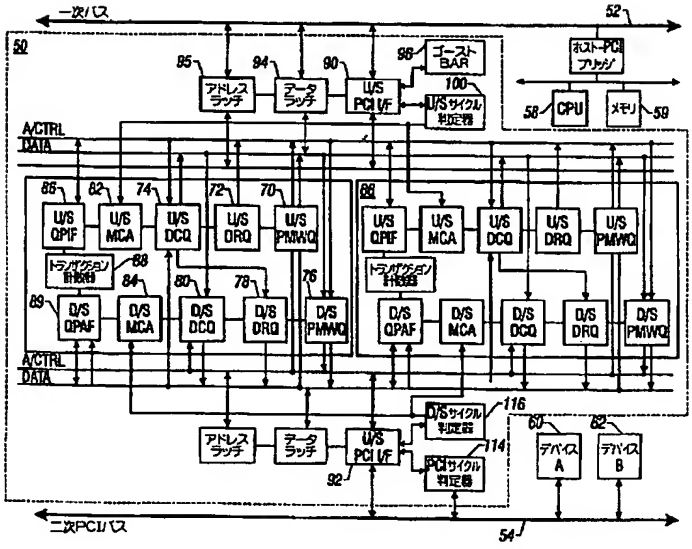
【図１】


CPU
10
主メモリ及び制御装置
12
ホスト バス
14
ホスト-PCI ブリッジ
18
一次PCIバス
16
PCI-PCI ブリッジ
20
二次PCIバス
19
22
24
26
PCIデバイス (ビデオカード)
PCIデバイス (NIC)
PCI-PCI ブリッジ
局所的PCIバス
30
28
32
PCIデバイス (SCSI)
PCI オプションカード

【図３】


200
202
トランザクション 実行待ち行列
トランザクション 命令待ち行列
TRQ 制御論理
MCA 制御論理
TDQ 制御論理
206
204
208

【図２】


一次バス
52
50
ホスト-PCI ブリッジ
CPU
メモリ
58
59
ゴースト BAR
96
100
U/S サイクル 判定器
アドレス ラッチ
データ ラッチ
U/S PCI I/F
A/CTRL
DATA
U/S QPIF
U/S MCA
U/S DCQ
U/S DRQ
U/S PMWQ
トランザクション 制御部
D/S QPAF
D/S MCA
D/S DCQ
D/S DRQ
D/S PMWQ
D/S サイクル 判定器
116
PCI サイクル 判定器
114
92
デバイス A
デバイス B
60
62
二次PCIバス
54

【図4】

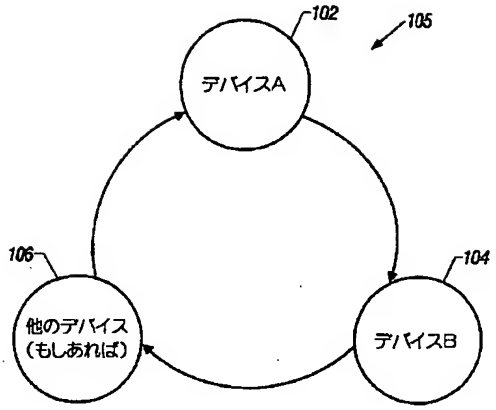


FIG. 4A

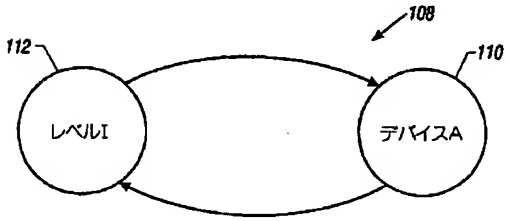
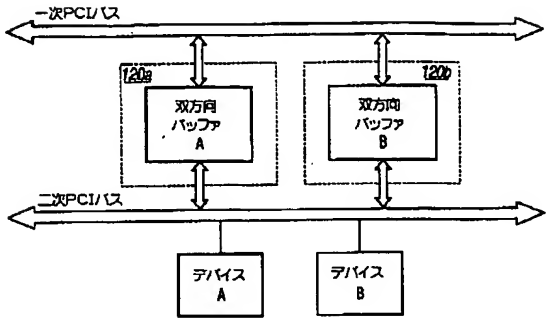


FIG. 4B

【図5】



フロントページの続き

(71)出願人 591030868
20555 State Highway
249, Houston, Texas
77070, United States o
f America

【外国語明細書】

1. Title of Invention

DELIVERING TRANSACTIONS BETWEEN DATA BUSES IN A COMPUTER SYSTEM

2. Claims

1. A computer system comprising:
a first device on a first data bus,
second and third devices on a second data bus, and
a bridge device interposed between the first and second data buses and configured to complete transactions between the first device and the second device without regard to whether the transactions entered the bridge device before or after transactions between the first device and the third device.

2. The computer system of claim 1 wherein the bridge device preserves the chronological order of the transactions between the first device and the second device.

3. The computer system of claim 1 wherein the bridge device attempts to accelerate completion of a delayed transaction between the first device and the second device when the delayed transaction is preventing completion of another transaction between the first and second devices.

4. The computer system of claim 3 wherein the other transaction comprises a downstream read transaction from a CPU.

5. The computer system of claim 1 further comprising a ghost base address register that aids in determining which of the devices on the second data bus is the target of a transaction initiated by the first device.

6. The computer system of claim 1 further comprising a grant signal that grants control of the second data bus to either the second or third device.

7. The computer system of claim 6 wherein the bridge device is configured to use the grant signal to determine whether the second or third device has initiated a transaction on the second data bus.

8. The computer system of claim 1 wherein one of the data buses comprises a PCI bus.

9. The computer system of claim 1 wherein each of the data buses comprises a PCI bus.

10. A method for use in controlling the delivery of two transactions between a device on a first data bus and each of two other devices on a second data bus across a bridge device serving the data buses, the method comprising:

storing the transactions in the bridge device, and

delivering one of the transactions without regard to the chronological order in which the transactions were stored in the bridge device.

11. The method of claim 10 further comprising thereafter delivering the other transaction.

12. The method of claim 10 further comprising preserving the chronological order of transactions to be delivered between the device on the first data bus and one of the other devices.

13. The method of claim 10 further comprising attempting to accelerate delivery of a delayed transaction between the device on the first data bus and one of the other devices when the delayed transaction is preventing another transaction from being delivered between the devices.

14. The method of claim 13 wherein the other transaction comprises a downstream read transaction from a CPU.

15. The method of claim 10 wherein one of the buses comprises a PCI bus.

16. The method of claim 10 wherein each of the buses comprises a PCI bus.

17. A computer system comprising:
a CPU on a primary PCI bus,
two PCI devices on a secondary PCI bus, and
a PCI-to-PCI bridge device interposed between the primary and secondary buses and comprising:
a storage area that stores two transactions, one to be delivered between the CPU and each of the PCI devices, and
an arbitration circuit that selects one of the transactions for delivery without regard to the chronological order in which the transactions were stored in the storage area.

18. The computer system of claim 17 wherein the arbitration circuit is configured to select the other transaction after selecting the one transaction.

19. The computer system of claim 17 wherein the storage area comprises two queue blocks, one of which stores transactions occurring between the CPU and one of the PCI devices, and the other of which stores transactions occurring between the CPU and the other PCI device.

20. The computer system of claim 17 wherein the storage area comprises:

a first set of buffers having one buffer that stores data flowing from the CPU to a first one of the PCI devices and another buffer that stores data flowing from the first PCI device to the CPU, and

a second set of buffers having one buffer that stores data flowing from the CPU to a second one of the PCI devices and another buffer that stores data flowing from the second PCI device to the CPU.

21. A bridge device for delivering data transactions between devices on two data buses in a computer system, the bridge device comprising:

for each pair of devices that may transact across the bridge device, a dedicated storage area that aids in completing transactions between the devices in the pair, and

a controller that allows transactions in one dedicated storage area to be completed without regard to the completion of earlier-issued transactions in another dedicated storage area.

22. The bridge device of claim 21 wherein each dedicated storage area comprises a portion dedicated to data flowing in a particular direction across the bridge device.

23. The bridge device of claim 22 wherein each dedicated storage area comprises another portion dedicated to data flowing in the opposite direction across the bridge device.

24. The bridge device of claim 21 wherein the dedicated storage area comprises a dedicated buffer for each type of transaction that may cross the bridge.

25. The bridge device of claim 24 wherein the transaction types include posted writes, delayed requests, and delayed completions.

3. Detailed Description of Invention

The invention relates to delivering transactions between data buses in a computer system.

Computer systems often include one or more data buses that allow components of the computer system to communicate. For example, one common type of data bus is a Peripheral Component Interface (PCI) bus, which provides a special communication protocol between the computer system's CPU/main memory and peripheral components, such as small computer system interface (SCSI) devices and network interface cards (NICs). When system memory and the peripheral components (e.g., PCI devices) reside on different buses, a bridge is required to manage the flow of data transactions between the two buses. Typically, system memory and PCI devices each reside, at least indirectly, on PCI buses connected by a PCI-to-PCI bridge. PCI bus architecture is defined by the PCI Local Bus Specification, Revision 2.1 ("PCI Spec 2.1"), published in June 1995, by the PCI Special Interest Group, Portland, Oregon, incorporated by reference. PCI-to-PCI bridge architecture is defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.0 ("PCI Bridge Spec 1.0"), published in April 1994, by the PCI Special Interest Group, also incorporated by reference.

Under the PCI Spec 2.1 and PCI Bridge Spec 1.0 standards, PCI bridges support two types of transactions: posted transactions (including all memory write cycles), which are completed on the initiating bus before they complete on the target bus, and delayed transactions (including memory read requests and I/O and configuration

read/write requests), which are completed on the target bus before they are completed on the initiating bus. According to the PCI Spec 2.1, PCI-to-PCI bridges must strongly favor posted write transactions when determining the order in which transactions are to be performed on the target bus. The PCI Spec 2.1 requires that, in order to prevent lock-up conditions, the computer system must allow posted write cycles to bypass earlier-initiated delayed request cycles and must prevent delayed request cycles from bypassing earlier-initiated posted write cycles. This requirement holds even when a posted write transaction and a delayed request transaction occur between two different pairs of devices.

Referring to Figure 1, a typical computer system includes a central processing unit (CPU) 10 and a main memory device 12 (including a memory controller) connected to a host bus 14. The host bus 14 communicates with a PCI bus 16 through a host-to-PCI bridge device 18. PCI bus 16 in turn communicates with another PCI bus 19 through a PCI-to-PCI bridge 20. PCI bus 16 is known as the "primary" PCI bus, and PCI bus 19 is known as the "secondary" PCI bus, because PCI bus 16 is closer hierarchically to the host bus 14 than is PCI bus 19.

Several peripheral devices, including video card 22, NIC 24, and PCI option card 26, are connected to the secondary PCI bus 19. Each device plugs in to a "slot" on the secondary PCI bus 19. The PCI option card 26 includes a PCI-to-PCI bridge device 28 and a PCI bus 30 of its own, which allows additional PCI devices to plug into a single slot on the secondary PCI bus 19. One or more PCI devices, such as SCSI device 32, may reside on the PCI option card 26. The PCI option card 26 operates as a "multi-threaded" device (i.e., a device that can maintain multiple delayed

transactions simultaneously) when more than one PCI device is connected to its local PCI bus 30.

Transactions initiated by the CPU and targeting a device on the secondary PCI bus 19 are known as "downstream" transactions, and transactions initiated by a device on the secondary bus 19 and targeting main memory 12 are known as "upstream" transactions. Transactions initiated by a device on the secondary bus 19 and targeting another device on the secondary bus 19 are known as "peer-to-peer" transactions. Peer-to-peer transactions are not dealt with here.

According to PCI Spec 2.1, any posted write transaction initiated on the primary or secondary PCI bus and posted in the PCI-to-PCI bridge device 20 must be completed on the target bus before any subsequently-issued transaction can be completed on the target bus. In addition, any transaction that completes on the initiating bus before a delayed read transaction is initiated on the bus must be completed on the target bus before the delayed read transaction is completed on the target bus. Likewise, any transaction that is completed on the initiating bus after a delayed read transaction completes on the initiating bus must complete after the delayed read transaction on the target bus. Therefore, a memory write transaction posted in the PCI-to-PCI bridge 20 by SCSI device 22 must be completed on the primary PCI bus 16 before a later-issued delayed read request from NIC 24 is completed, and the posted memory write transaction must invalidate prefetch data associated with any earlier-issued delayed read transactions from NIC 24 (i.e., the prefetch data must be flushed from the PCI-to-PCI bridge 20 when the memory write transaction is posted).

In one aspect, the invention features a computer system having a first device on a first data bus, second and third devices on a second data bus, and a bridge device interposed between the first and second data buses and configured to complete transactions between the first device and the second device without regard to whether the transactions entered the bridge device before or after transactions between the first device and the third device.

Embodiments of the ivention may include one or more of the following features. The bridge device may preserve the chronological order of the transactions between the first device and the second device. The bridge device may attempt to accelerate completion of a delayed transaction between the first device and the second device when the delayed transaction is preventing completion of another transaction (e.g., a downstream read transaction from the CPU) between the first and second devices. The computer system may include a ghost base address register that aids in determining which of the devices on the second data bus is the target of a transaction initiated by the first device. The computer system also may include a grant signal that grants control of the second data bus to either the second or third device, and the bridge device may be configured to use the grant signal to determine whether the second or third device has initiated a transaction on the second data bus. One or both of the data buses may be a PCI bus.

In another aspect, the invention features a method for use in controlling the delivery of two transactions between a device on a first data bus and each of two other devices on a second data bus across a bridge device serving the data buses. The transactions first are stored in the

bridge device, and then one of the transactions is delivered without regard to the chronological order in which the transactions were stored in the bridge device. The other transaction then may be delivered.

Embodiments of the invention may include one or more of the following features. The chronological order of transactions to be delivered between the device on the first data bus and one of the other devices may be preserved. Delivery of a delayed transaction between the device on the first data bus and one of the other devices may be accelerated when the delayed transaction is preventing another transaction (e.g., a downstream read transaction from the CPU) from being delivered between the devices. One or both of the buses may be a PCI bus.

In another aspect, the invention features a computer system having a CPU on a primary PCI bus, two PCI devices on a secondary PCI bus, and a PCI-to-PCI bridge device interposed between the primary and secondary buses. The bridge device includes a storage area that stores two transactions, one of which is to be delivered between the CPU and each of the PCI devices, and an arbitration circuit that selects one of the transactions for delivery without regard to the chronological order in which the transactions were stored in the storage area.

Embodiments of the invention may include one or more of the following features. The arbitration circuit may be configured to select the other transaction after selecting the one transaction. The storage area may include two queue blocks, one of which stores transactions occurring between the CPU and one of the PCI devices, and the other of which stores transactions occurring between the CPU and the other PCI device. The storage area may include a first set of buffers having one buffer that stores data flowing from the

CPU to a first one of the PCI devices and another buffer that stores data flowing from the first PCI device to the CPU. The storage area also may include a second set of buffers having one buffer that stores data flowing from the CPU to a second one of the PCI devices and another buffer that stores data flowing from the second PCI device to the CPU.

In another aspect, the invention features a bridge device for delivering data transactions between devices on two data buses in a computer system. For each pair of devices that may transact across the bridge device, the bridge device includes a dedicated storage area that aids in completing transactions between the devices in the pair. The bridge devivce also includes a controller that allows transactions in one dedicated storage area to be completed without regard to the completion of earlier-issued transactions in another dedicated storage area.

Embodiments of the invention may include one or more of the following features. Each dedicated storage area may include a portion dedicated to data flowing in a particular direction across the bridge device, and it also may include another portion dedicated to data flowing in the opposite direction across the bridge device. The dedicated storage area may include a dedicated buffer for each type of transaction that may cross the bridge. The transaction types may include posted writes, delayed requests, and delayed completions.

Among the advantages of the embodiments of the invention are the following. Transaction ordering rules may be relaxed so that transactions between two devices are ordered strictly only with respect to other transactions between the same two devices. For example, in a delayed transaction architecture, a delayed read transaction between

a CPU and a first PCI device may be completed before an earlier-issued posted write transaction between the CPU and another PCI device. In addition, prefetch data associated with an earlier-issued delayed read transaction between the CPU and the first PCI device need not be invalidated when the write transaction between the CPU and the other PCI device is posted. Therefore, computer systems relying on PCI bridges may operate more efficiently since data delivered during delayed transactions is not discarded as frequently.

Other features and advantages will become apparent from the following description and from the claims.

Description of Specific Preferred Embodiments of the Invention

Referring to Figure 2, a computer system 50 includes a PCI-to-PCI bridge device 50 (PPB or "bridge") connecting primary and secondary PCI buses 52 and 54, respectively, that follows relaxed transaction ordering rules. In particular, the bridge 50 operates such that transactions between CPU 58 and PCI device 60 ("Device A") are ordered only with respect to other transactions between these two devices, without regard to the chronological relationship

between these transactions and transactions occurring between CPU 58 and PCI device 62 ("Device B").

The bridge 50 achieves relaxed transaction ordering by maintaining two identical sets of buffers 64 and 66, or "queue blocks," for each PCI device "pair" that may transact across the bridge 50. For example, queue block 64 temporarily stores transactions occurring between CPU 58 and Device A 60, while queue block 66 temporarily stores transactions occurring between CPU 58 and Device B 62. The bridge 50 will include an additional queue block for every other PCI device on the secondary PCI bus 54. The bridge device 50 treats CPU 58 and memory 59 as a single device on the primary PCI bus 52. The buffers in the queue blocks 64 and 66 need not be permanently assigned to the secondary PCI slots, but instead may be dynamically assigned, e.g., at system start-up.

Each queue block 64 and 66 manages transactions flowing between the primary PCI bus 52 and the devices 60 and 62 on the secondary PCI bus 54. The queue blocks 64 and 66 are similar, so only queue block 64 is described below. Queue block 66 (and any other queue blocks that may be in the bridge device 52) function similarly. Each queue block resembles the queue block shown and described in U.S. patent application 08/655,254, filed June 5, 1996, by John MacLaren and Alan Goodrum and incorporated by reference in its entirety.

Each queue block 64 contains three upstream queues 70, 72, and 74, which store information flowing upstream (i.e., from the corresponding PCI device 60 on the secondary PCI bus 54 to the CPU 58), and three downstream queues 76, 78, and 80, which store information flowing downstream (i.e., from the CPU 58 to the corresponding PCI device 60). The queue block 64 also includes two cycle arbiters 82 and

84 and two queue block-to-PCI (QPIF) interfaces 86 and 89, which govern the flow of information out of either the upstream or downstream queues, as described below.

The first downstream queue, a posted memory write queue 76, stores posted memory write cycles issued by the CPU 58 on the primary bus 52 along with all information required to execute each cycle on the secondary bus 54. The PMWQ 76 has four transaction buffers, each of which holds one posted memory write transaction containing up to eight cache lines (256 bytes) of data. The upstream PMWQ 70 stores posted memory write cycles issued by PCI device 60 on the secondary bus 54 and functions similarly to the downstream PMWQ 76. The PMWQs 70 and 76 are described in more detail in U.S. patent application 08/655,254, filed June 5, 1996.

The second downstream queue, a delayed request queue (DRQ) 78, stores delayed request transactions (i.e., delayed read requests (DRR), such as memory read (MR), memory read line (MRL), and memory read multiple (MRM) requests; and input/output (I/O) read/writes and configuration (config) read/writes) issued by the CPU 58 on the primary bus 52, along with all information required to execute each transaction on the secondary bus 54. The DRQ 78 has three transaction buffers, each of which is capable of holding one double-word, or "dword," of data for delayed writes. Likewise, the upstream DRQ 72 stores delayed request transactions issued by PCI device 60 on the secondary bus 64 and functions similarly to the downstream DRQ 78. The DRQs 72 and 78 are described in more detail in U.S. patent application 08/655,254, filed June 5, 1996.

The third downstream queue, a delayed completion queue (DCQ) 80, stores delayed completion information provided by the CPU 58 in response to delayed request

transactions generated by PCI device 60. For a delayed read request, the corresponding completion information contains the read data requested by the initiating device and the read status (i.e., an indication of whether a parity error or target abort occurred). The delayed completion information returned for a delayed write transaction is the same as that returned for a delayed read request, except that no data is returned for delayed writes.

The DCQ 80 has eight completion buffers, each of which can hold up to eight cache lines of completion information for a single delayed request. In addition to the completion information, each completion buffer also stores a copy of the delayed request that generated the information. This copy of the delayed request is used to determine when a subsequent request by the issuing PCI device is a retry of the stored request. If a subsequent request matches the stored request and the completion buffer contains the requested data, the DCQ 80 provides the completion data to the requesting device.

Similarly, the upstream DCQ 74 stores delayed completion information provided by PCI device 60 in response to delayed requests from the CPU 58. Because I/O and config read/writes occur only on the downstream bus, only the upstream DCQ 74 may contain delayed completion information corresponding to one of these transactions.

If one of the DCQs 74 or 80 determines that one of its buffers contains data intended for a requesting device but different than the data requested in the current transaction, the buffer may be flushed to prevent the requesting master from receiving stale data. The buffer is flushed when it contains prefetch data (i.e., data left in the buffer after the requesting device has retrieved some of the data, or data that was not specifically requested by the

device), but is not flushed when it contains completion data (i.e., data specifically requested by a device that has not yet returned to retrieve it). If the buffer contains completion data and the requesting device has issued a request that does not "hit" the buffer, the DCQ 144 tags the device as a "multi-threaded" device (i.e., one that is capable of maintaining more than one transaction at once) and allocates another completion buffer for the new request.

Each buffer in the DCQs 74 and 80 progresses through several states as data passes through. The buffer is in an "empty" state when it contains no data, a "complete" state when it contains data specifically requested by the initiating device, and a "prefetch" state when it contains data that was not specifically requested but was prefetched by the bridge device 52 (e.g., after the initiating device retrieved all requested data from the buffer and kept its "frame_" signal asserted, indicating that it wanted more data). The DCQs 74 and 80 are described in more detail in U.S. patent application 08/655,254, filed June 5, 1996.

Each master cycle arbiter (MCA) 82 and 84 in the queue block 64 maintains ordering constraints between posted memory write, delayed request, and delayed completion transactions in either the downstream queues or the upstream queues. The relaxed ordering constraints are similar to those set forth in the PCI Bridge Architecture Specification, Version 2.1, except that strict ordering is maintained only between transactions involving the same pair of devices (i.e., transactions between the CPU 58 and PCI device 60 are ordered only with respect to other transactions between those two devices). These ordering constraints require that bus cycles maintain strong write ordering and that deadlocks do not occur. Therefore, each MCA 82 or 84 determines the order in which posted memory

write transactions in the corresponding PMWQ 70 or 76 and delayed request transactions in the corresponding DRQ 72 or 78 are run on the corresponding PCI bus 52 or 54. Each MCA 82 or 84 also controls the availability of delayed completion information stored in the corresponding DCQ 74 or 80. To ensure compliance with these rules, the downstream MCA 84 gives each posted memory write cycle an opportunity to bypass earlier-issued delayed request cycles, while both the downstream and the upstream MCAs 84 and 82, respectively, do not allow delayed request and delayed completion cycles to bypass earlier-issued posted memory write cycles.

Referring also to Figure 3, each MCA 82 or 84 uses two transaction queues, a transaction run queue (TRQ) (or transaction execution queue) 200 and a transaction order queue (TOQ) 202, to manage cycles enqueued in the corresponding PMWQ, DRQ, and DCQ. An MCA control block 204 receives transactions from the PMWQ, DRQ, and DCQ and outputs run commands. The transactions are moved into and out of the TRQ 200 and TOQ 202 by a TRQ control block 206 and a TOQ control block 208, respectively.

The TRQ 200 is the queue from which the MCA determines the transaction execution order. Transactions in the TRQ 200 can be executed in any order without violating the transaction ordering rules, but once a posted memory write cycle is placed in the TRQ 200, no other cycle can be placed in the TRQ 200 until the posted memory write is removed. Transactions in the TRQ 200 are tried in circular order and generally are completed in the order in which they were received. However, if a transaction in the TRQ 200 is retried on the PCI bus, the MCA 82 or 84 may select the next transaction in the TRQ 200 to be tried on the PCI bus. Delayed completion transactions are never placed in the TRQ

200 because they are slave cycles rather than master cycles. Furthermore, because delayed completion information may be made available to the requesting device as soon as it enters the DCQ if no posted memory write cycles are pending in the PMWQ, delayed completion transactions are placed in the TOQ 202 only when a posted memory write cycle is pending in the TRQ 200.

The TRQ 200 is a circular queue that holds up to four transactions at once. Because the MCA always must be able to run at least one posted memory write transaction to preserve the required ordering constraints, the TRQ 200 never can hold more than three delayed request transactions at once. Furthermore, the TRQ can hold only one posted write transaction at a time because posted writes cannot be passed by any later-initiated transaction, including other posted writes.

The TOQ 202 is a first-in-first-out (FIFO) queue that retains the historical order of transactions received by the bridge after a posted memory write transaction is placed in the TRQ 200. Because all transactions must wait for earlier-issued posted memory writes to run, all transactions including posted memory write, delayed request, and delayed completion transactions, are placed in the TOQ 202 when a posted memory write is enqueued in the TRQ 200. Transactions in the TOQ 202 must remain in the TOQ 202 until the posted memory write transaction is removed from the TRQ 200. The TOQ 202 can hold up to three posted memory write transactions (the fourth will be stored in the TRQ 200), three delayed request transactions, and four delayed completion transactions. The MCAs 82 and 84, including TRQ 200, TOQ 202, and control blocks 204, 206, and 208 are described in more detail in U.S. patent application 08/655,254, filed June 5, 1996.

An upstream queue-to-PCI interface (QPIF) 86 manages transactions flowing from the upstream queues 70, 72, and 74 to the primary PCI bus 52, and from the primary PCI bus 52 to the downstream DCQ 80. The QPIF 86 enters a "master" mode to run posted memory write and delayed request transactions from the PMWQ 70 and the DRQ 72 on the primary bus 52. The QPIF 86 enters a "slave" mode to provide data from the upstream DCQ 74 to the CPU 58 or to send transactions from the primary PCI bus 52 to the downstream DCQ 80.

When the QPIF 86 receives a posted write transaction from the bus 52 it forwards the transaction to the downstream PMWQ 76 if a corresponding one of a group of transaction counters 88 indicates that the downstream PMWQ 76 is not full. When the QPIF 86 receives a delayed request, it first forwards the request to the DCQ 74 to determine whether the transaction already has been placed in the upstream DCQ 74 and, if so, whether the corresponding delayed completion information has been returned by PCI device 60. If the delayed completion information is in the upstream DCQ 74, the information is provided to the CPU 58 and the transaction is terminated. If the request already is enqueued but the delayed completion information has not been returned, the CPU 58 is retried and the transaction is terminated on the primary PCI bus 52. If the transaction is not yet enqueued, the upstream DCQ 74 reserves a completion buffer for the transaction and the QPIF 86 forwards the transaction to the downstream DRQ 78, as long as the corresponding one of the transaction counters 88 indicates that the downstream DRQ 78 is not full. The downstream QPIF 89 functions similarly to the upstream QPIF 86. The QPIFs 86 and 89 are described in more detail in U.S. patent application 08/655,254, filed June 5, 1996.

The transaction counters 88 maintain a count of the number of transactions enqueued in the upstream and downstream queues, respectively. A posted memory write (PMW) counter indicates the number of PMW transactions held in the corresponding posted memory write queue. The PMW counter is incremented each time a PMW transaction is sent to the corresponding PMWQ. The counter is decremented each time a PMW cycle has been completed on the corresponding PCI bus. Likewise, a delayed request (DR) counter counts the number of DR transactions held in the corresponding delayed request queue. A delayed completion (DC) counter counts the number of delayed completions that are enqueued in the corresponding master cycle arbiter. The transaction counters are described in more detail in U.S. patent application 08/655,254, filed June 5, 1996.

The bridge device 52 includes upstream and downstream PCI interface devices 90 and 92, respectively. The upstream PCI interface 90 allows the bridge device 50 to communicate with the devices (i.e., the CPU 58 and main memory 59) on the primary bus 52, and the downstream PCI interface 92 allows the bridge device 50 to communicate with the devices (i.e., Device A 60 and Device B 62) on the secondary bus 54.

When the CPU takes control of the primary bus and initiates a transaction, the upstream PCI interface 90 becomes a PCI slave device. The interface 90 latches the PCI command, address, and data in data and address latches 94 and 95. The interface 90 then determines which of the downstream PCI devices 60 and 62 is the target of the transaction. To do so, the interface compares the memory address provided by the CPU 58 to the addresses in ghost base address registers (Ghost BARs) 96 in the bridge device 50. (The Ghost BARs 96 are created by snooping the base

address registers (BARs) created by POST at start-up. Each pair of BARs maps a PCI slot to a block of memory, and the Ghost BARs contain copies of these mappings.) The PCI interface 90 assumes that the CPU 58 is the master of all downstream transactions.

Referring also to Figure 4A, an upstream cycle arbiter 100 uses a "Level I" round robin scheme 105 to select the queue block 64 or 66 from which an upstream transaction should be run on the primary bus 52. The arbiter 100 first selects the queue block 64 corresponding to Device A 60 (state 102). The arbiter 100 then selects the queue 66 corresponding to Device B 62, and if any other PCI devices are connected to the secondary bus 54, the arbiter 100 progressively selects the queues corresponding to those devices. If one of the queue blocks 64 and 66 does not contain a transaction to run, the arbiter 100 skips the corresponding state in its round robin scheme 105. If the transaction from the selected queue block is retried, the arbiter 100 moves to the next state in the round robin scheme 105.

Referring to Figure 4B, the PCI Spec 2.1 contains a "bridge interface priority" rule providing that when both the primary and secondary buses initiate a transaction simultaneously, the arbiter 100 must give priority to the downstream transaction. But in some bridges that allow write posting, this requirement may lead to a deadlock if a downstream device posts a write at the same time that the CPU issues a read request to the downstream device. In this situation, the bridge 50 would be instructed both to complete the posted write before completing any delayed requests for the same device and to complete the downstream read before completing any transactions issued simultaneously by a downstream master.

To prevent a deadlock from occurring, the upstream cycle arbiter in some embodiments may need to switch to a "Level II" arbitration scheme 108 if this condition occurs. In the Level II scheme 108, the arbiter 100 elevates the priority of the simultaneously-issued posted write transaction in an attempt to complete the transaction more quickly (state 110). After the upstream interface 90 tries to run the posted write transaction on the primary bus 52, the arbiter 100 returns to the "Level I" arbitration scheme 105 of Figure 4A to select the next transaction to run. If the posted write transaction did not run successfully on the primary bus 52, the arbiter 100 returns to the Level II arbitration scheme 108 after selecting a transaction from the Level I scheme 105 (i.e., the arbiter 100 continues in the Level II scheme 108 until the posted write transaction successfully completes).

Referring again to Figure 2, the downstream PCI interface 92 operates similarly to the upstream PCI interface 90, with a few exceptions. First, unlike the upstream cycle arbiter 100, the downstream cycle arbiter 116 never will need to use a two-level arbitration scheme, because the downstream PCI interface 92 is not subject to the "bridge interface priority" rule. Second, the downstream interface 92 always assumes that the CPU 58 is the target device and does not need to rely on the Ghost BARs 96 to determine which PCI device 60 or 62 is the master. Instead, the downstream interface 92 monitors the REQ and GNT lines in the PCI arbiter 114 to determine which of the devices 60 and 62 on the secondary PCI bus 54 is the master and therefore which queue block 64 or 66 should receive the transaction.

Other embodiments are within the scope of the following claims. For example, referring to Figure 5, the

PCI-to-PCI bridge device may include a set of bidirectional transaction queues 120a, 120b for each PCI device pair instead of two sets of unidirectional queues per pair. The invention also may be implemented in non-PCI architectures and in PCI-architectures that do not follow PCI Spec. 2.1 or PCI Bridge Spec. 1.0.

4. Brief Description of Drawings

Figure 1 is a block diagram of a computer system.

Figure 2 is a block diagram of a computer system in which data buses are connected by a bridge that implements relaxed transaction ordering rules.

Figure 3 is a block diagram of a master cycle arbiter that assures proper ordering of transactions.

Figure 4 includes figures 4A and 4B, figures 4A and 4B are state transiton diagrams of an arbitration shema.

Figure 5 is a block diagram of an alternative bridge architecture.

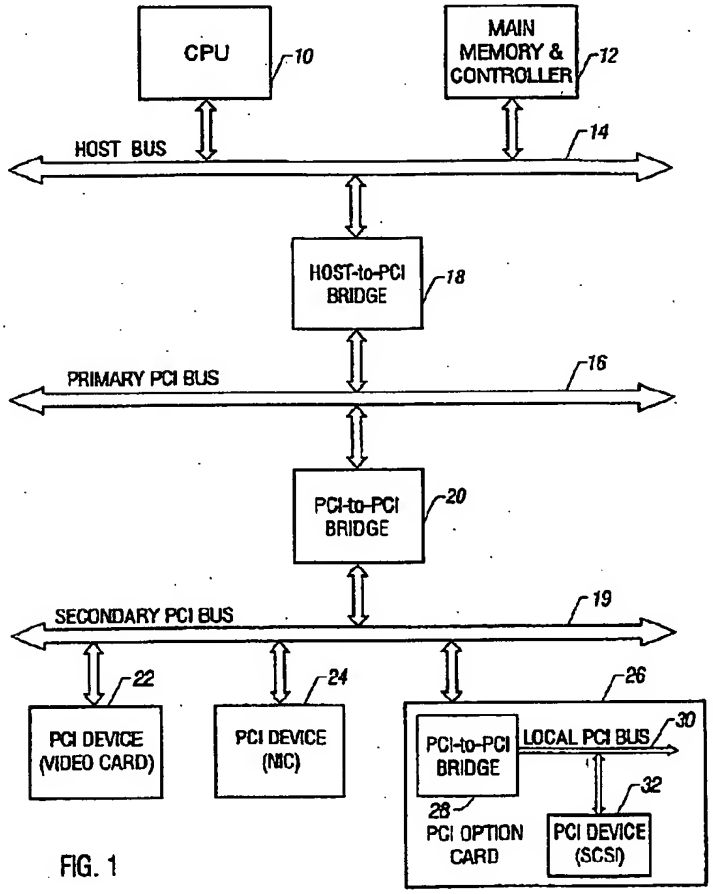

CPU
10
MAIN MEMORY & CONTROLLER
12
HOST BUS
14
HOST-to-PCI BRIDGE
18
PRIMARY PCI BUS
16
PCI-to-PCI BRIDGE
20
SECONDARY PCI BUS
19
22
24
26
PCI DEVICE (VIDEO CARD)
PCI DEVICE (NIC)
PCI-to-PCI BRIDGE
LOCAL PCI BUS
30
32
28
PCI OPTION CARD
PCI DEVICE (SCSI)

FIG. 1

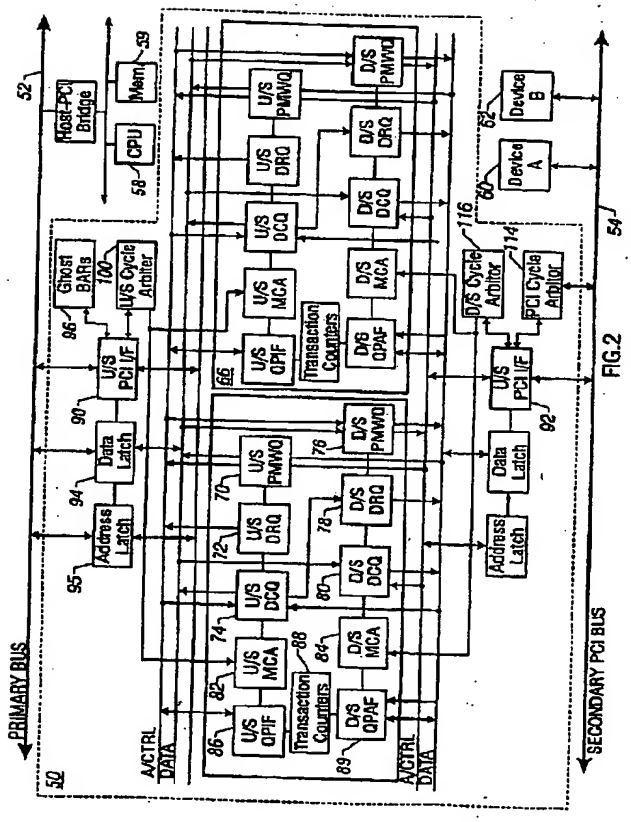

PRIMARY BUS
SECONDARY PCI BUS
Host-PCI Bridge
CPU
Mem
Ghost BARs
U/S Cycle Arbiter
D/S Cycle Arbitor
PCI Cycle Arbitor
U/S PCI I/F
D/S PCI I/F
Data Latch
Address Latch
U/S PMWQ
U/S DRQ
U/S DCQ
U/S MCA
U/S QPIF
D/S PMWQ
D/S DRQ
D/S DCQ
D/S MCA
D/S QPAF
Transaction Counters
A/CTRL
DATA
Device A
Device B

FIG.2

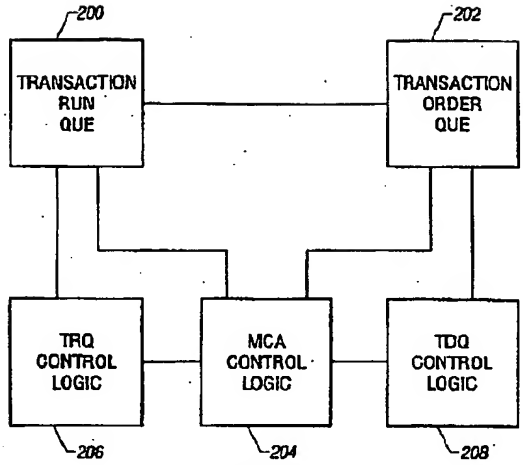
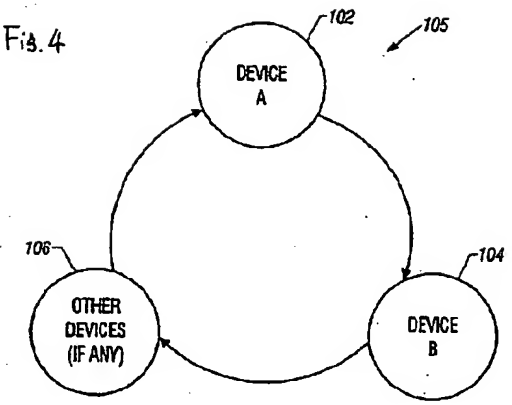
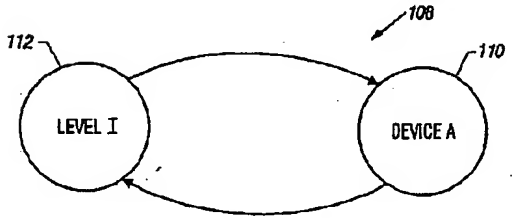

200
202
TRANSACTION RUN QUE
TRANSACTION ORDER QUE
TRQ CONTROL LOGIC
MCA CONTROL LOGIC
TDQ CONTROL LOGIC
206
204
208

FIG. 3

Fig. 4


102
105
DEVICE A
106
104
OTHER DEVICES (IF ANY)
DEVICE B

(4A)


108
112
110
LEVEL I
DEVICE A

(4B)

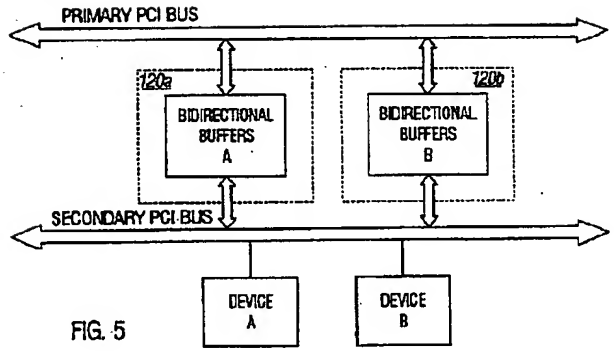

PRIMARY PCI BUS
120a
BIDIRECTIONAL BUFFERS A
120b
BIDIRECTIONAL BUFFERS B
SECONDARY PCI BUS
DEVICE A
DEVICE B

FIG. 5

1 Abstract

A bridge device for delivering data transactions between devices on two data buses in a computer system includes, for each pair of devices that may transact across the bridge device, a dedicated storage area that aids in completing transactions between the devices in the pair. The bridge device also includes a controller that allows transactions in one dedicated storage area to be completed without regard to the completion of earlier-issued transactions in another dedicated storage area.

2 Representative Drawing

Fig. 1.

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成17年8月4日(2005.8.4)

【公開番号】特開平10-301893
【公開日】平成10年11月13日(1998.11.13)
【出願番号】特願平9-370468
【国際特許分類第７版】
　Ｇ０６Ｆ　13/36
【ＦＩ】
　Ｇ０６Ｆ　13/36　　３１０Ｆ
　Ｇ０６Ｆ　13/36　　３１０Ｃ

【手続補正書】
【提出日】平成16年12月27日(2004.12.27)
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
２つのデータバスであって、一方のデータバスは、該データバスの１つを介して少なくとも１つの他のデータ処理デバイスにデータ書き込み又はデータを読出す要求を、少なくとも１つのデータ処理デバイスから受け取るように構成された該２つのデータバスと、
該要求を該データバスの１つに配置するようにしているデータ処理デバイスに許可信号を発生する調停デバイスと、
該データバス間に挿入されたブリッジデバイスであって、順序スキームを実行するように構成され、該順序スキームは該ブリッジデバイスに、書き込み要求および次の要求をその順序で１つのデータバスから受け取り、そして次の要求および書き込み要求をその順序で他のデータバスに送る、該ブリッジデバイスと、
を備え、該ブリッジデバイスが、どのデータデバイスが該データバスに要求を配置したかを決定するために許可信号を使用するように構成されたコンピュータシステム。
【請求項２】
前記順序スキームが前記ブリッジデバイスに、前記１つのデータバスが１つのデータ処理デバイスから両方の要求を受け取った場合だけ、前記書き込み要求の送達前に、次の要求の送達を可能にすることを特徴とする請求項１のコンピュータシステム。
【請求項３】
前記順序スキームが前記ブリッジデバイスに、前記書き込み要求が前記次の要求の送達を妨げている時に、該書き込み要求の送達を加速する試みを可能にすることを特徴とする請求項１のコンピュータシステム。
【請求項４】
前記次の要求がＣＰＵからのダウンストリーム読出し要求を含むことを特徴とする請求項３のコンピュータシステム。
【請求項５】
そこから読み出され、また、そこへ書き込みされる前記他のデータバスにおけるデータ処理デバイスを示す情報を含むゴーストベースアドレスレジスタを更に備えた請求項１のコンピュータシステム。